neoMOTE

Crossbow in collaboration with MEMSIC

EcoWizard/neoKIT
データアクセス説明書

2010 年 11 月 18 日 Rev.6

◆住友精密工業株式会社

改定履歴

Rev	日付	内容
0	2008-11-29	作成
1	2009-01-07	4ch-Pt100 ノードの温度変換式の誤りを修正
2	2009-01-23	board_id の誤りを修正
3	2009-06-02	公開
4	2009-12-21	EcoWizard の内容を追記
5	2010-03-31	・ 変換式の誤り修正（16bit ADC のフルスケール値 65536→65535） ・ 各フィールドの型を追記 ・ 基地局ノード直接通信時の注意事項追記 ・ CRC 方式について追記
6	2010-11-18	・ パルス入力ノード(高速パルス対応)を追記 ・ アナログ入力ノードを追記

目次

1. はじめに 2
1.1. 本ドキュメントの目的 2
1.2. EcoWizard と neoKIT の違いについて 2
1.3. 参考文献 2
2. 概要 3
2.1. EcoWizard/neoKIT データフロー 3
2.1.1. データフロー概念図 3
2.1.2. アクセス方法 3
3. 基地局ノードから直接データを受信する 4
3.1. 物理的インタフェース 4
3.1.1. USB 4
3.1.2. RS232C 4
3.1.3. シリアルポート通信パラメータ 4
3.2. シリアルフレームプロトコル 5
3.2.1. 概略 5
3.2.2. フレームフォーマット 5
3.2.3. エスケープ処理 5
3.2.4. CRC 6
3.3. センサノードパケットフォーマット 6
3.3.1. TOS_Msg ヘッダ 6
3.3.2. XMesh ヘッダ 7
3.3.3. neoKIT センサノードデータ 7
3.3.4. EcoWizard センサノードデータ 9
3.4. 基地局ノードから直接データを受信する場合の注意事項 14
3.4.1. 基地局との通信状態確認 14
3.4.2. 各ノードとの通信状態確認 14
3.4.3. 重複データの除去 14
3.4.4. データ受信タイミング 14
4. XServe からソケット経由でデータを受信する 15
4.1. ソケットインタフェース 15
4.2. データフォーマット 16
5. SQLite データベースにアクセスしてデータを取り出す 18
5.1. SQLite データベースファイル 18
5.2. sqlite3 コマンドによるデータ取り出し 19
5.2.1. sqlite3 コマンドインタラクティブモード使用 19
5.2.2. 出力ファイル、モードの指定 19
5.2.3. SQL 文によるデータの取り出し 19
5.2.4. さまざまなデータの取り出し方（条件指定など） 20
5.2.5. 出力ファイル 20
5.3. sqlite ライブラリによるデータ取り出し 20

1. はじめに

1.1. 本ドキュメントの目的

本ドキュメントは EcoWizard/neoKIT のセンサデータにユーザ機器等から独自にアクセスするための方法について説明するものです。

ソフトウエアによるアクセス以外にもユーザが手動でデータを取り出す方法についてもカバーしています。

1.2. EcoWizard と neoKIT の違いについて

EcoWizard（エネルギー監視システム）と neoKIT（汎用センサネットワークシステム）はデータアクセス手順という観点からみるとほぼ同じものです。したがって本ドキュメントは両製品について説明しますが、以下の違いがありますのでご注意ください。

- ゲートウェイのハードウエア：EcoWizard は UbuntuPC、neoKIT は NB100 です。
- クライアントソフトウエア：EcoWizard は EcoWizard、neoKIT は MoteExplorer です。
- 基地局の USB シリアルの通信速度：EcoWizard は 115200bps、neoKIT は 57600bps です。
- センサノード種別：「3.3 センサノードパケットフォーマット」にて説明します。

1.3. 参考文献

- neoKIT-J 取扱説明書
- EcoWizard 取扱説明書
- EcoWizard サーバ管理マニュアル

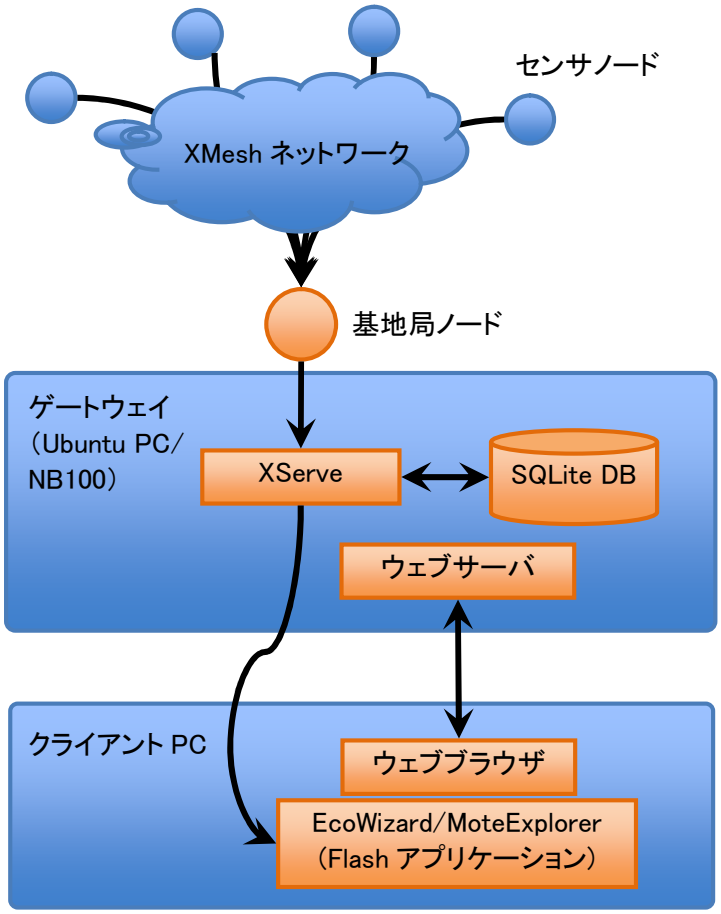
2. 概要

本章ではEcoWizard/neoKITのデータフローとそれらにアクセスする方法の概要を説明します。

2.1. EcoWizard/neoKIT データフロー

2.1.1. データフロー概念図

EcoWizard/neoKIT のデータフロー概念図を以下に示します。



この図からわかるように、複数のセンサノードから収集されたセンサデータは最終的にはウェブブラウザ内で実行される Flash アプリケーションである EcoWizard/MoteExplorer を通して、ユーザが利用できるようになっています。

バックエンドのデータベースエンジンとして SQLite が、センサネットワークからのパケットデータを処理するミドルウエアとして XServe が、UbuntuPC/NB100 ゲートウェイ内で使われています。

2.1.2. アクセス方法

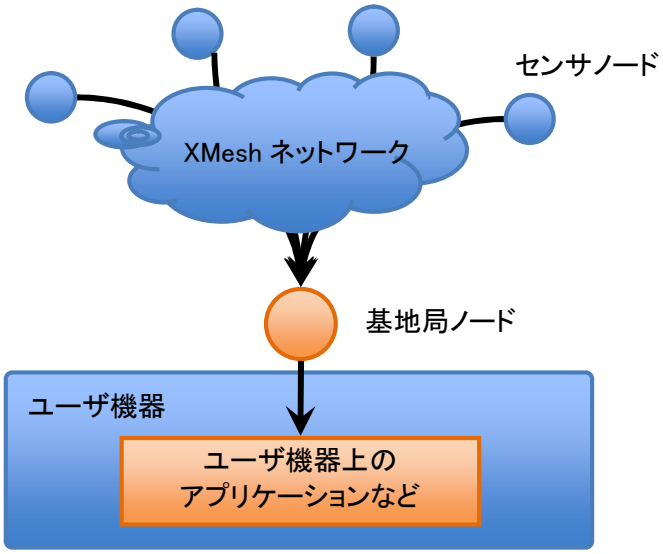
データフローのユーザ側の終端であるウェブアプリケーション以外の場所でユーザ機器等が独自にセンサデータにアクセスする方法は以下のとおり複数あります。

- 基地局ノードから直接データを受信する
- ゲートウェイ上の XServe プログラムからソケット経由でデータを受信する
- ゲートウェイ上の SQLite データベースにアクセスしてデータを取り出す

次章以降で、これらについてより詳しく説明していきます。

3. 基地局ノードから直接データを受信する

本章では基地局ノードをユーザ機器に接続して、基地局ノードから送られてくるセンサデータを直接受信する方法について説明します。



3.1. 物理的インタフェース

基地局ノードとユーザ機器との物理的な接続インタフェースは USB と RS232C の二つの方法があります。

3.1.1. USB

基地局ノード(WB-Z101B-K/X)あるいは、メンテナンスノード(WM-Z1100-)に基地局ノード用ソフトウエアを書き込んで USB インタフェースボード(MIB520)に接続したものと USB 経由で接続する形態です。

いずれの場合でもユーザ機器側に FTDI 社の FT2232C[1]用の仮想 COM ポートドライバがインストールされている必要があります。このドライバによって基地局ノードはユーザ機器上ではシリアルポートとしてみえます。

なお本形態ではユーザ機器の USB ポートが USB バスパワー経由で neoMOTE 側に給電できる必要があります。基地局側の消費電力は約 250mW(50mA/5.0V)です。

3.1.2. RS232C

メンテナンスノード(WM-Z1100-)に基地局ノード用ソフトウエアを書き込んで RS232C インタフェースボード(MIB510)経由で接続する形態です。

3.1.3. シリアルポート通信パラメータ

USB、RS232C いずれの接続方法でも、ユーザ機器からはシリアルポート経由でデータにアクセスできます。シリアルポートの通信パラメータは以下の通りです。

[1] http://www.ftdichip.com/FTProducts.htm#FT2232C

項目	値	
ボーレート	EcoWizard: neoKIT:	115200bps 57600bps
スタートビット	1	
ストップビット	1	
データビット	8	
パリティ	なし	
フロー制御	なし	

3.2. シリアルフレームプロトコル

3.2.1. 概略

前述した物理的インタフェースを通じて、ユーザ機器はシリアルポートとして基地局ノードと通信を行うことができます。ここではこの通信に使われるシリアルフレームプロトコルについて説明します。

基地局ノードは各センサノードからパケットを受信するたびにシリアルポート側へデータを転送します。したがってユーザ機器側からはコマンドを送信することなくシリアルポートを読めば順次パケットを受信することができます。ただしパケット受信頻度は各センサノードのセンシング間隔やセンサノードの数によって変動します。ユーザ機器側ではデータ受信までブロックしないシリアルポートからの読み出しを行うか、あるいは別のスレッドで読み出し処理を行うか、などする必要があります。

3.2.2. フレームフォーマット

基地局ノードがシリアルポートへパケットを送信する際に使うフレームフォーマットは以下のとおりです。

- まずフレームの区切りを示す SYNC_BYTE(0x7E)がフレームの先頭と末尾にあります。先頭の SYNC_BYTE に続いてフレームオプションを示す Type フィールドが続きます。EcoWizard/neoKIT においては常に 0x42 です。
- これに続いて現れるのがセンサノードからのパケットであり、これは TOS_Msg ヘッダ部と XMesh ヘッダ部、データ内容部の 3 つに分けられます。
- センサノードからのパケットの後ろにはフレームの整合性チェックのための CRC フィールドが 2 バイトつけられています。

Sync 0x7E	**Type** 0x42	**Data** TOS_Msg ヘッダ	**Data** XMesh ヘッダ	**Data** データ内容	**CRC** (16bit)	**Sync** 0x7E

3.2.3. エスケープ処理

データ部には任意の値(0x00～0xFF)が使われますが、SYNC_BYTE(0x7E)と同じものがあった場合にはフレームの終了と区別がつきません。そこで以下のようにエスケープ処理されます。

- エスケープ文字は 0x7D です。
- データ中の 0x7E あるいは 0x7D は、エスケープ文字に続けてそのバイトを 0x20 で XOR したものが置かれます。例えば 0x7E は 0x7D,0x5E になり、0x7D は 0x7D,0x5D になります。

したがってユーザ機器側では上記のエスケープ処理の逆を行って、もとのデータに戻す必要があります。

3.2.4. CRC

CRC はパケットが壊れていないかチェックするためのものです。方式は CRC-16-CCITT です。ユーザ機器でフレームを受信した際には下記のようにして CRC をチェックしてください。

- Type フィールド(0x42)から Data フィールド(TOS_Msg ヘッダ、XMesh ヘッダ、データ内容)までを以下の CRC 計算関数で 1 バイトずつ計算してください。
- CRC の初期値は 0 です。
- エスケープ処理された部分に関しては、復元した 1 バイトを計算に使ってください。

```
int xcrc_byte(int crc, byte b) {
    byte i;

    crc = crc ^ ((int)b) << 8;
    i = 8;
    for (i = 8; i > 0; i--) {
        if ((crc & 0x8000) != 0){
            crc = crc << 1 ^ 0x1021;
        }else{
            crc = crc << 1;
        }
    }
    return crc;
}
```

3.3. センサノードパケットフォーマット

ここではセンサノードのパケット部分のフォーマットについて説明します。1 バイトより長いデータについては little-endian 形式で格納されています。つまり下位バイトから順に並べられます。

表中の offset はデータ部の先頭からのオフセットバイト数を示します。

3.3.1. TOS_Msg ヘッダ

すべてのデータパケットはこの TOS_Msg ヘッダからはじまります。

フィールド名	説明	型	長さ	offset
addr	パケットの宛先。ユーザ機器に渡されるものは常に 0x007E	uint16	2	0
amtype	パケットタイプ センサデータを処理するためにはこのフィールドが 0x0B か 0x0D のパケットのみ処理してください。	uint8	1	2
group	グループ ID	uint8	1	3
length	以下に続く XMesh ヘッダ、センサノードデータ部の長さ CRC 部は含まない	uint8	1	4

3.3.2. XMesh ヘッダ

TOS_Msg ヘッダの type フィールドが 0x0B か 0x0D の場合には、XMesh ヘッダが続きます。

フィールド名	説明	型	長さ	offset
sourceaddr	このパケットを直前に送信したノード ID ユーザ機器に渡されるものでは常に 0	uint16	2	5
nodeid	このパケットを送信した元のノード ID	uint16	2	7
seqno	パケットの通し番号	uint16	2	9
socketid	アプリケーション・サービス識別用の番号 常に 0x33	uint8	1	11

3.3.3. neoKIT センサノードデータ

XMesh ヘッダに続いてはセンサデータ部が続きます。データは生データ(ADC カウント値など)が入っているので変換式を使って実際の計測データに変換してください。

各センサノードデータに共通のバッテリ電圧については下記の変換式を使って生データ S を電圧 V(mV)に変換してください。バッテリ電圧については EcoWizard/neoKIT 共通です。

$$V = \frac{1252352}{S}$$

温湿照度センサノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x31	uint8	1	12
packet_id	パケットバージョン。常に 0	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
temp	温度	uint16	2	18
humidity	湿度	uint16	2	20
light	照度	uint16	2	22
seq_no	データパケットの通し番号	uint16	2	24

温度の変換式は次式です。生データ S を温度 T(℃)に変換してください。

$$T = \left(S \div 100\right) - 100$$

湿度の変換式は次式です。生データ S を湿度 R(%)に変換してください。

$$R = S \div 100$$

照度の変換式は次式です。まず生データ S を電圧 V(V)に変換し、それを使って照度 L(lux)に変換してください。

$$V = \frac{2.5 \times S}{65535 \times 2} \qquad L = 10^{3 \times \frac{V}{0.465}}$$

4ch-Pt100 センサノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x30	uint8	1	12
packet_id	パケットバージョン。常に 0	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
temp0	温度 0ch	uint16	2	18
temp1	温度 1ch	uint16	2	20
temp2	温度 2ch	uint16	2	22
temp3	温度 3ch	uint16	2	24
seq_no	データパケットの通し番号	uint16	2	26

温度の変換式は次式です。生データ S を温度 T(℃)に変換してください。

$$T = \frac{-A + \sqrt{A^2 - 4B \times \left(1 - \frac{2.38419 \times 10^{-3} \times S}{100}\right)}}{2B}$$

$$A = 3.9083 \times 10^{-3}$$

$$B = -5.775 \times 10^{-7}$$

4ch 電圧測定ノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x33	uint8	1	12
packet_id	パケットバージョン。常に 0	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
vol0	電圧 0ch	uint16	2	18
vol1	電圧 1ch	uint16	2	20
vol2	電圧 2ch	uint16	2	22
vol3	電圧 3ch	uint16	2	24
seq_no	データパケットの通し番号	uint16	2	26

電圧の変換式は次式です。生データ S を電圧 V(mV)に変換してください。

$$V = \frac{2.5 \times 1000 \times S}{65535}$$

中継ノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x34	uint8	1	12
packet_id	パケットバージョン。常に 0	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
seq_no	データパケットの通し番号	uint16	2	18

3.3.4. EcoWizard センサノードデータ

XMesh ヘッダに続いてはセンサデータ部が続きます。データは生データ(ADC カウント値など)が入っているので変換式を使って実際の計測データに変換してください。

各センサノードデータに共通のバッテリ電圧についてはneoKITと同じですので、そちらを参照ください。

電力量センサノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x83	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
unitID	ユニット番号。常に 1	uint8	1	18
circuitNum	回路番号。1～8	uint8	1	19
totalPower1	トータル積算電力(kWh)の 100 倍値	uint32	4	20
totalPower2	トータル積算電力(単相 2 線 T 側)(kWh)の 100 倍値	uint32	4	24
voltageRN	電圧 RN(3 相 3 線の場合 RS)(V)の 10 倍値	uint32	4	28
voltageRT	電圧 RT(3 相 3 線の場合 ST)(V)の 10 倍値	uint32	4	32
voltageTN	電圧 TN(3 相 3 線の場合 TR)(V)の 10 倍値	uint32	4	36
currentR	電流 R(A)の 100 倍値	uint32	4	40
currentS	電流 S(A)の 100 倍値	uint32	4	44
currentT	電流 T(A)の 100 倍値	uint32	4	48
momentPower1	瞬時電力(kW)の 1000 倍値	uint32	4	52
momentPower2	瞬時電力(単相 2 線 T 側)(kW)の 1000 倍値	uint32	4	56
powerFactor1	力率(%)の 10 倍値	uint16	2	60
powerFactor2	力率(単相 2 線 T 側)(%)の 10 倍値	uint16	2	62
seq_no	データパケットの通し番号	uint16	2	64

パケット中では整数表現としているため、実際の値の何倍かになっている項目があるので注意してください。

パルス入力ノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x81	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
channel	チャンネル番号。0～3	uint8	1	18
pulselevel	パルスレベル。bit0 にチャンネル 0、bit1 にチャンネル 1、bit2 にチャンネル 2、bit3 にチャンネル 3 のパルスレベルがビットで格納されている。	uint16	2	19
pulse	積算パルス量	uint16	2	21
seq_no	データパケットの通し番号	uint16	2	23

パルス入力ノード（高速パルス対応）

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x8D	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
channel	チャンネル番号。0～3	uint8	1	18
pulselevel	パルスレベル。bit0 にチャンネル 0、bit1 にチャンネル 1、bit2 にチャンネル 2、bit3 にチャンネル 3 のパルスレベルがビットで格納されている。	uint16	2	19
pulse	積算パルス量	uint32	4	21
seq_no	データパケットの通し番号	uint16	2	25

アナログ入力ノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x8C	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
current1	電流 Ch1	uint16	2	18
current2	電流 Ch2	uint16	2	20
current3	電流 Ch3	uint16	2	22
voltage1	電圧 Ch1	uint16	2	24
voltage2	電圧 Ch2	uint16	2	26
voltage3	電圧 Ch3	uint16	2	28
seq_no	データパケットの通し番号	uint16	2	30

電流の変換式は次式です。生データ S を電流(mA)に変換してください。

$$A[mA] = \frac{S}{65535} \times 2.5 \times \frac{1}{82} \times 10^{-3}$$
$$= \frac{S}{65535} \times 3.0488 \times 10^{-5}$$

電圧の変換式は次式です。生データ S を電圧(V)に変換してください。

$$V[V] = \frac{S}{65535} \times 2.5 \times \left\{ \frac{62 \times 10^{3} + \left(\frac{1}{\frac{1}{39 \times 10^{3}} + \frac{1}{10 \times 10^{6}}} \right)}{\left(\frac{1}{\frac{1}{39 \times 10^{3}} + \frac{1}{10 \times 10^{6}}} \right)} \right\}$$

$$= \frac{S}{65535} \times 6.4899$$

電流センサノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x80	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
channel	チャンネル。0～3	uint8	1	18
range	CT 定格電流（A）	uint16	2	19
average	平均電流	uint16	2	21
max	最大電流	uint16	2	23
min	最小電流	uint16	2	25
integration	トータル積算電流	uint32	4	27
overflow	オーバーフローが発生したかどうかのフラグ。1 で発生	uint8	1	28
seq_no	データパケットの通し番号	uint16	2	32

平均電流・最大電流・最小電流の変換式は次式です。生データ S を電流(A)に変換してください。

$$A = \frac{1.2 \times R \times S}{65535}$$

R : CT定格電流

トータル積算電流の変換式は次式です。生データ S を積算電流(Ah)に変換してください。

$$A = \frac{1.2 \times R \times S}{65535 \times 3600}$$

R : CT定格電流

温湿照度センサノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x86	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
temp	温度	uint16	2	18
humidity	湿度	uint16	2	20
light	照度	uint16	2	22
seq_no	データパケットの通し番号	uint16	2	24

温度の変換式は次式です。生データ S を温度 T(℃)に変換してください。

$$T = (S \div 100) - 100$$

湿度の変換式は次式です。生データ S を湿度 R(%)に変換してください。

$$R = S \div 100$$

照度はパケット内にルクス(lx)値に変換された値が入っていますので変換は不要です。

4ch-Pt100 センサノード

EcoWizard 用の 4ch-Pt100 センサノードのパケットフォーマットは neoKIT と同じですので、そちらを参照ください。

中継ノード

フィールド名	説明	型	長さ	offset
board_id	パケット識別子。常に 0x35	uint8	1	12
packet_id	パケットバージョン。常に 0x85	uint8	1	13
parent_id	親ノード ID	uint16	2	14
voltage	バッテリ電圧	uint16	2	16
seq_no	データパケットの通し番号	uint16	2	18

3.4. 基地局ノードから直接データを受信する場合の注意事項

3.4.1. 基地局との通信状態確認

ユーザ機器と基地局との通信は、以下のような理由により正常に行えないことがあります。

- ケーブルの抜け・断線
- 基地局ノードのインタフェースボードからの脱落、インタフェースボードの電源切断
- 基地局ノード・インタフェースボードの故障

高信頼性が必要なシステムにおいては、以下に述べる方法でユーザ機器と基地局ノードとの接続状況の確認を行うことを推奨します。

基地局ノードは 5 秒（EcoWizard 基地局、または neoKIT HP 基地局）あるいは 60 秒（neoKIT LP 基地局）間隔で Heartbeat と呼ばれる、接続確認用パケットを送信し続けています。ユーザ機器側でこの間隔を基準に一定時間以内に何らかのパケットが受信できているかどうかを確認してください。

Heartbeat パケットの構造は以下のとおりです。

フィールド名	説明	型	長さ	offset
addr	常に 0x007E	uint16	2	0
amtype	常に 0xfd（=253）	uint8	1	2
group	グループ ID	uint8	1	3
length	常に 2	uint8	1	4
round	0～0xffff（=65535）をインクリメントし続けるカウンタ	uint16	2	5

受信を確認するパケットは Heartbeat パケットだけにしないでください。これはセンサデータパケットと競合した場合には Heartbeat パケットがユーザ機器に送信されない場合もあるからです。その場合はかわりにデータパケットが送信されますので、何らかのパケットが上記 Heartbeat 送信間隔以内[2]に受信できれば通信は行えていると確認できます。

3.4.2. 各ノードとの通信状態確認

特定のノードからのパケットの欠落がシステム上大きな問題となる場合、そのノードとの通信状態を確認するようにしてください。具体的には、そのノードからのパケットが、ノードのデータ送信間隔に対して一定期間以上届いていない場合にノードが異常（電源切断あるいは電波環境の悪化など）であるとみなして警報を発するような対応が考えられます。

3.4.3. 重複データの除去

同じセンサデータパケットが連続して送信される場合があります。これは、実際には基地局でパケットを受信できたにもかかわらず受信確認パケットが送信元のセンサノードに届かず、そのセンサノードがパケットを再送した場合に起こります。

これを回避するには、各センサデータパケットにある「データの通し番号」というフィールドを使ってください。これが同じものは重複データとみなせます。ただし、このフィールドは overflow した後 0 に戻りますのでご注意ください。

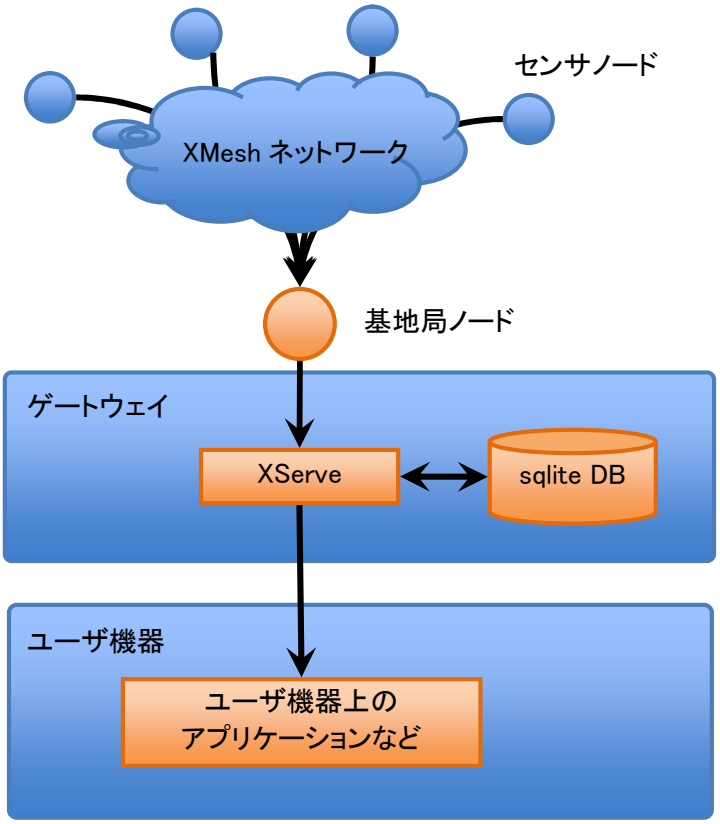
3.4.4. データ受信タイミング

センサデータパケット受信タイミングは各センサノードからの自発的な送信に依存します。したがって非常に短い間隔（数ミリ秒）で連続して複数ノードからのデータパケットが基地局から送信されることもあります。

[2] 実際には Heartbeat 送信間隔の 2～3 倍とマージンを持たせることを推奨します。

4. XServe からソケット経由でデータを受信する

本章ではゲートウェイ(UbuntuPC/NB100)上で実行されている XServe からソケット通信によってデータを受信する方法について説明します。



4.1. ソケットインタフェース

ゲートウェイ(UbuntuPC/NB100)がユーザ機器から LAN 経由でアクセス可能な状態で、以下のポートにアクセスすることによって後述する XML ストリームデータを読み出すことができます。

項目	値
IP アドレス	ゲートウェイ(UbuntuPC/NB100)の IP アドレスまたは ecowizard-####/Sluggo-#####
ポート	9005

4.2. データフォーマット

前述のポートから読み出される XML ストリームは以下のフォーマットをしています。

長さ	XML ストリーム
4bytes、little-endian	<?xml version="1.0" ?><MotePacket>～</MotePacket>

先頭に 4 バイト、後に続く XML ストリームの長さ(バイト数)が little-endian で格納されているので、まず 4 バイトを読み出してから次にそのバイト数分だけ後に続く XML ストリームを読み出すようにしてください。

XML ストリーム本体の要素のツリー構造は以下の通りです。便宜上改行を入れていますが、実際には改行されていません。

```
<?xml version=”1.0” ?>
<MotePacket>
        <PacketName>
                パケット名
        </PacketName>
        <ParsedDataElement>
                <Name>
                        フィールド名
                </Name>
                <ConvertedValue>
                        センサ値(変換済)
                </ConvertedValue>
                <ConvertedValueType>
                        センサ値の型
                </ConvertedValueType>
        </ParsedDataElement>
        ...(ParsedDataElement)が繰り返される
</MotePacket>
```

実際には内部的に使用される要素も入っていますが、センサ値をユーザ機器から扱うという目的であれば以下のような処理で十分でしょう。

1. XML ストリーム読出し
2. XML 宣言のチェック
3. MotePacket 要素中の ParsedDataElement 要素毎に Name 要素のフィールド名が目的のものに合致するかチェックする
4. 合致した場合同じ ParsedDataElement 要素内の ConvertedValue 要素から変換済のセンサ値を得る

以下に実際の 4ch 電圧測定ノードの出力例を示します。便宜上改行を入れていますが、実際には改行されていません。

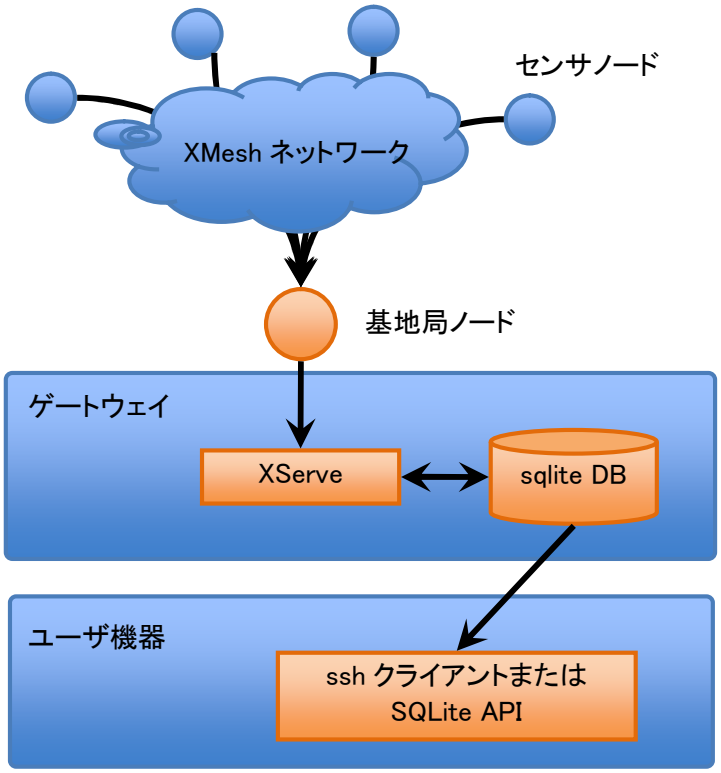
```
<?xml version="1.0" ?>
<MotePacket>
        <PacketName>PhaseII MultiChannelVolApp Multihop Config XML</PacketName>
        <ParsedDataElement>
                <Name>amtype</Name>
                <ConvertedValue>13</ConvertedValue>
                <ConvertedValueType>uint8</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
```

```
                <Name>nodeid</Name>
                <SpecialType>nodeid</SpecialType>
                <ConvertedValue>121</ConvertedValue>
                <ConvertedValueType>uint16</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>parent</Name>
                <ConvertedValue>0</ConvertedValue>
                <ConvertedValueType>uint16</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>group</Name>
                <ConvertedValue>160</ConvertedValue>
                <ConvertedValueType>uint8</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>socketid</Name>
                <ConvertedValue>51</ConvertedValue>
                <ConvertedValueType>uint8</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>board_id</Name>
                <SpecialType>sensorboardid</SpecialType>
                <ConvertedValue>51</ConvertedValue>
                <ConvertedValueType>uint8</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>packet_id</Name>
                <ConvertedValue>0</ConvertedValue>
                <ConvertedValueType>uint8</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>voltage</Name>
                <ConvertedValue>3.278408</ConvertedValue>
                <ConvertedValueType>float</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>vol0</Name>
                <ConvertedValue>1280.250488</ConvertedValue>
                <ConvertedValueType>float</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>vol1</Name>
                <ConvertedValue>1280.174316</ConvertedValue>
                <ConvertedValueType>float</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>vol2</Name>
                <ConvertedValue>1280.212402</ConvertedValue>
                <ConvertedValueType>float</ConvertedValueType>
        </ParsedDataElement>
        <ParsedDataElement>
                <Name>vol3</Name>
                <ConvertedValue>1280.250488</ConvertedValue>
                <ConvertedValueType>float</ConvertedValueType>
        </ParsedDataElement>
        <internal>
                <nodeId>121</nodeId>
                <sensorTable>sci21vol_results</sensorTable>
        </internal>
</MotePacket>
```

5. SQLite データベースにアクセスしてデータを取り出す

本章ではセンサデータが格納されている、ゲートウェイ(UbuntuPC/NB100)内の SQLite データベースにアクセスしてデータを取り出す方法について説明します。

なお、EcoWizard については「EcoWizard サーバ管理マニュアル」を参照ください。



5.1. SQLite データベースファイル

SQLite データベースファイルは以下の場所に置かれています(NB100 内のパス)。

```
/usr/xserve/sys/xserve_sqlite.db
```

また Windows のファイル共有経由で NB100 のフォルダにアクセスすることができます。詳細については neoKTI-J 取扱説明書の「無線センサネットワークをつなぐ」の「NB100 を接続する」の項を参照ください。

この場合、NB100 内のパスの /usr/xserve がファイル共有によって公開されますので、SQLite データベースファイルは以下のパスでユーザ機器からアクセスできます[3]。

```
¥¥sluggo-#####¥xserve¥sys¥xserve_sqlite.db
```

[3] #####の部分には NB100 のシリアル番号が入ります。

5.2. sqlite3 コマンドによるデータ取り出し

手動でデータベースからデータを取り出すには NB100 にログインして sqlite3 コマンドを使います。

5.2.1. sqlite3 コマンドインタラクティブモード使用

詳細は neoKIT-J 取扱説明書の「SQLite データベースの操作」の「SQLite データベース」「SQLite データベースのエクスポート」の項を参照してください。

ssh クライアントソフトから NB100 に root アカウントでログインします（ここでは ssh コマンドを利用）。パスワードは `sluggomote` です。

```
$ ssh root@sluggo-#####
```

出力ファイルを置くディレクトリへ移動して（ここでは/usr/xserve とした）、データベースファイルを引数に指定して sqlite3 コマンドを実行してください。

```
sluggo-#####:~# cd /usr/xserve/
sluggo-#####:/usr/xserve# bin/sqlite3 sys/xserve_sqlite.db
SQLite version 3.3.16
Enter ".help" for instructions
sqlite>
```

プロンプトが `sqlite>` になっていればインタラクティブモードになっているということです。

このインタラクティブモードではコマンドは.（ピリオド）ではじめることにより、データベース操作を行う SQL 文と区別しています。.quit でインタラクティブモードを終了しますが、間違ったコマンドを打つなどして SQL 入力モードのままになった場合などは Ctrl キー+D で終了させることもできます。

5.2.2. 出力ファイル、モードの指定

続けて出力ファイル名を指定し、Excel で読み出せるように CSV モードに設定します。なお neoKIT-J 取扱説明書では`.mode` を使用せずデフォルトの'|'区切りを使用しています。

また`.header` コマンドにより出力データにフィールド名のヘッダを付加するかどうかを指定できます。

```
sqlite> .output test.csv
sqlite> .mode csv
sqlite> .header ON
```

5.2.3. SQL 文によるデータの取り出し

sqlite データベースから前項で指定した出力ファイルへデータを出力させるには SQL 文を実行します。本ドキュメントでは SQL の詳しい説明までは行いません。一般的な SQL の解説書あるいはウェブ上のリファレンス等を参照ください。SQL 文の最後には;（セミコロン）をつけることを忘れないようにしてください。

SQL 文で指定するテーブル名は以下の通りです。なおインタラクティブモードでは`.tables` コマンドですべてのテーブル名を、`.schema` でテーブル名を指定するとそのテーブルのフィールド名や型を表示できます。

センサノード	テーブル名
温湿照度センサノード	`scs22_results`
4ch-Pt100 センサノード	`sci21_results`
4ch 電圧測定ノード	`sci21vol_results`
中継ノード	`emrouter_results`

フィールド名については3.3.3の表にあるフィールド名を参照してください。この表にない、すべてのテーブルが持っているフィールドに result_time があります。これはセンサデータがデータベースに格納された時間を表します。

EcoWizard については「EcoWizard サーバ管理マニュアル」を参照ください。

5.2.4. さまざまなデータの取り出し方（条件指定など）

以下にさまざまなデータの取り出し方の例を示します。

温湿照度センサノードのデータをすべて出力します

```
select * from scs22_results;
```

4ch 電圧測定ノードの電圧 ch0 だけを出力します

```
select result_time,nodeid,vol0 from sci21vol_results;
```

4ch 電圧測定ノードの 2008 年 11 月 23 日から 25 日までのデータを出力します（25 日のデータは含まない）。

```
select * from sci21vol_results where result_time between "2008-11-23
" and "2008-11-25";
```

4ch 電圧測定ノードの 2008 年 11 月 23 日 0 時 0 分から 1 時 0 分までのデータを出力します。

```
select * from sci21vol_results where result_time between "2008-11-23
00:00:00" and "2008-11-23 01:00:00";
```

5.2.5. 出力ファイル

SQL 文を実行したら、あらかじめ .output コマンドで指定しておいた出力ファイルにデータがテキストで出力されています。先の例の場合は¥¥sluggo-#####¥xserve¥test.csv ファイルが出力ファイルなのでこれをコピーして Excel などで開くだけでデータの後処理を行うことができます。

5.3. sqlite ライブラリによるデータ取り出し

SQLite にはデータベースにアクセスするためのライブラリが用意されています。これを使えばユーザ機器上のプログラムから NB100 の SQLite データベースにアクセスすることができます。

ただし、将来的なデータベース構造の変更などに伴ってユーザ機器側のプログラムを逐一変更しなければいけないことや、使用するライブラリ特有の問題により正常にアクセスできない可能性があること、などからこの方法については推奨はされません。

既知の問題として NB100 の SQLite データベース中の float フィールドの形式は Windows などの SQLite ライブラリが想定している形式と異なっているため正常に読み出せません[4]。

[4] 上位と下位の 32bit を入れ替える必要がある